24-780 Engineering Computation Problem Set 11

You need to create a ZIP file (It may appear as a compressed folder in Windows) and submit the ZIP file via the 24-780 Canvas. The file name of the ZIP file must be:

PS11-YourAndrewID.zip

For example, if your Andrew account is *hummingbird@andrew.cmu.edu*, the file name must be:

PS11-hummingbird.zip

If your ZIP file does not comply with this naming rule, you will automatically lose 5% credit from this assignment. If we are not able to identify who submitted the file, you will lose another 5% credit. If we finally are not able to connect you and the submitted ZIP file, you will receive 0 point for this assignment. Therefore, please make sure you strictly adhere to this naming rule before submitting a file.

The ZIP file needs to be submitted to the 24-780 Canvas. If you find a mistake in the previous submission, you can re-submit the ZIP file with no penalty as long as it is before the submission deadline.

Notice that the grade will be given to the final submission only. If you submit multiple files, the earlier version will be discarded. Therefore, if you re-submit a ZIP file, the ZIP file MUST include all the required files. Also, if your final version is submitted after the submission deadline, late-submission policy will be applied no matter how early your earlier version was submitted.

Make sure you upload your Zip file to the correct location. If you did not upload your assignment to the correct location, you will lose 5%.

The ZIP file needs to include:

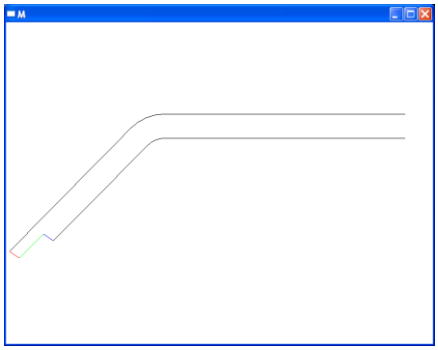
- C++ source file of your program (ps11.cpp)

Submission Due: 11/14 (Tue) 23:59

START EARLY!

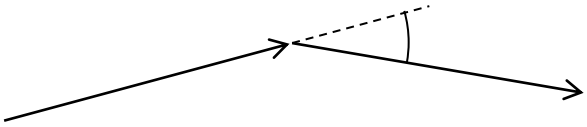
Unless you are already a good programmer, there is no way to finish the assignment overnight.

PS11 Segmentation of line segments

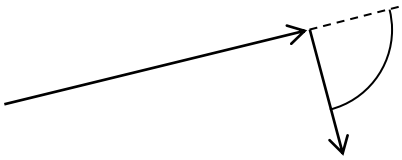


A designer draws cross sections of an instrument panel in automobile interior design. Such drawings often need to be segmented (or grouped) for further processing.

A very primitive way, but used very often, is segmenting the lines based on the bending angle. If two lines sharing a point bend at a small angle, two lines are in the same group. If such two lines bend at a large angle, two lines are in different groups. (An exception is a loop with only one sharp corner, in which case there will be only one group.)



Small bending: Same group



Large bending: Differnet groups

In this assignment, you write a program that splits a sequence of lines into multiple groups.

Suggested steps (but you don't have to follow if you come up with an easier way, as long as your program gives the same output on the console and graphics window.)

(1) Start from the base code

(2) Fill MeasureCornerAngle function so that it can measure an angle at a corner of line segments. Line segments is represented as an array of double in this program. X and Y coordinates of *i*th point (zero based) is coord[i*2] and coord[i*2+1] This function must return 0.0 if the given corner index is smaller than 1 or greater than the number of points-2. If you successfully fill this function, you will see the following output on the console window.

```
Corner 1  Angle 5.625001
Corner 2  Angle 11.249998
Corner 3  Angle 11.250001
Corner 4  Angle 11.250000
Corner 5  Angle 5.931727
Corner 6  Angle 100.943270
Corner 7  Angle 78.749999
Corner 8  Angle 78.750001
Corner 9  Angle 78.947684
Corner 10  Angle 5.822681
Corner 11  Angle 11.250004
Corner 12  Angle 11.249996
Corner 13  Angle 11.250004
Corner 14  Angle 5.624998
```

(3) Fill DrawLineSegments function. This function must change the color of lines at every corner where the corner angle is greater than 60 deg. The color sequence must be black, red, green, blue, and repeat this sequence after the fourth color. If you implement this function successfully, your line segments appear to be grouped into sub-groups. Your graphics window will look like as follows.

(Base code)

```
#include <stdio.h>
#include <math.h>
#include "fssimplewindow.h"

const double YsPi=3.1415927;

double MeasureCornerAngle(int nCoord,const double coord[],int index)
{
    return 0.0;
}

void DrawLineSegments(int nCoord,const double coord[])
{
}

int main(void)
{
    const int nCoord=16;
    const double coord[nCoord*2]=
    {
        747.000000, 171.000000,
        297.000000, 171.000000,
        279.441871, 172.729325,
        262.558491, 177.850842,
        246.998679, 186.167735,
        233.360390, 197.360390,
          6.167735, 426.998679,
         24.875801, 439.499009,
         69.875801, 394.499009,
         88.583867, 406.999340,
        265.180195, 229.180195,
        271.999340, 223.583867,
        279.779246, 219.425421,
        288.220936, 216.864662,
        297.000000, 216.000000,
        747.000000, 216.000000
    };

    for(int i=1; i<nCoord-1; i++)
    {
        printf("Corner %d  Angle %lf\n",i,MeasureCornerAngle(nCoord,coord,i)*180.0/YsPi);
    }

    FsOpenWindow(16,16,800,600,1);

    int key=FSKEY_NULL;
    while(FSKEY_ESC!=key)
    {
        FsPollDevice();
        key=FsInkey();

        glClear(GL_DEPTH_BUFFER_BIT|GL_COLOR_BUFFER_BIT);
        DrawLineSegments(nCoord,coord);
        FsSwapBuffers();

        FsSleep(20);
    }

    return 0;
}
```